Asymptotic Analysis of Three Way Merge Sort

Proof by Mathematical Induction:

$$recursion: 3T\left(\frac{n}{3}\right)$$

$$merge: O(n)$$

$$T(n) = 3T\left(\frac{n}{3}\right) + O(n)$$

$$Assume\ n = 3^k$$

$$T(3^k) = 3T\left(3^{k-1}\right) + O(3^k)$$

$$Therefore:$$

$$S(k) = T(3^k)$$

$$S(0) = O(1)$$

$$S(k) = 3S(k-1) + O(3^k)$$

$$recursion\ and\ substitution: S(k-1) = 3S(k-2) + O\left(3^{k-1}\right)$$

$$S(k) = 3^2 S(k-2) + 2O(3^k)$$

$$S(k) = 3^k S(k-k) + kO(3^k)$$

$$S(k) = 3^k O(1) + O(k3^k)$$

$$Substitute\ and\ simplify\ and\ ignore\ 3^k O(1)$$

$$T(n) = O\left(n \cdot \log_3 n\right)$$

$$Because\ we\ are\ only\ concerned\ with\ tail\ end\ behavior\ we\ can\ ignore\ base\ number$$

$$T(n) = O(n \cdot \log n)$$

We know this to be true because each tier of recursion produces a height of the tree log(n), as well as, each new tier will have the same amount of operations classified as n. Therefore the time complexity is n*log(n).

Specifically for this assignment, in the merge function of code we have seven while loops that represent c'n operations. The mergeSortThreeWayRecursion function of code is represented by c'log(n). These two functions dominate the overall sorting algorithm making the time complexity n*log(n).